In this document a **graph** refers to an **undirected finite graph**, meaning that it has finitely many vertices and edges, and that edges have no direction. Also a **weighted graph** refers to a graph where every edge has a non-negative weight.

Definition 1 (Minimum spanning tree (MST)).

For any connected weighted graph G:
Define a **spanning subgraph** of G to be a subgraph of G that contains all the vertices of G.
Define an **spanning tree** of G to be a spanning subgraph of G that is a tree.
Define an **MST** of G to be a spanning tree of G with minimum total weight.

Theorem 2 (Existence of MST).

For any connected weighted graph G:
G has an MST.

Proof.

For any connected weighted graph G:
G is a connected spanning subgraph of G.
Let T be a connected spanning subgraph with the fewest edges [by well-ordering].
If T has some cycle C:
Let T' be the same as T except with one edge removed from C.
Then T' is a connected spanning subgraph, contradicting the choice of T.
Therefore T has no cycle and hence is a spanning tree.
Thus G has an MST [because the number of spanning trees of G is finite and at least 1].

Theorem 3 (Spanning tree fact).

For any connected weighted graph G with n vertices:
Any connected spanning subgraph of G with $(n-1)$ edges is a tree.

Proof.

For any connected weighted graph G with n vertices:
For any connected spanning subgraph H of G with $(n-1)$ edges:
Let T be an MST of H [by Existence of MST].
Then T has $(n-1)$ edges [because T is a tree].
Thus $H = T$ [because T is a subgraph of H and they have the same number of edges].
Thus H is a tree.

Algorithm 4 (Kruskal's algorithm).

Procedure Kruskal(connected weighted graph G):
Set T to be a graph with the same vertices as G but no edges.
For each edge e in G in order of increasing weight:
If $T+e$ has no cycle:
Set $T = T+e$.
Return T.

Before we give the proof of Kruskal's algorithm, here is the intuitive idea. Instead of trying to prove that the final output is correct, we prove the invariance that at each step the graph T that the algorithm is constructing is part of some correct answer. The hard part is of course to prove that for each step, if before that step T is part of some correct answer, then after that step it is still part of some correct answer. This is where we use a 'swapping' argument, which goes as follows.

First we let T' be some correct answer that contains T. Then we see whether T' agrees with what we do in the current step. If it disagrees, we find a way to modify T' such that it still remains correct but agrees with us. Almost always this involves a swapping argument. This general technique for proving an algorithm's correctness works for many common optimization algorithms that you have seen, including Dijkstra's, Prim's and Kruskal's.

In the case of Kruskal's algorithm, a correct answer is an MST, and the existence of T' at the start is guaranteed by Existence of MST. At each step we want to add an edge e. If T' contains e, and the invariance is preserved. If T' does not contain e, it must have some other path to connect the endpoints of e, but that path cannot be completely contained in T otherwise $T+e$ would have a cycle and we would not have wanted to add e in the first place. So some edge e' on that path is not in T.

If e' has lower weight than e, we would have considered e' earlier in the algorithm and would have added it, because both T and e' are contained in T', and T' has no cycle. But this contradicts our choice of e' as an edge not in T.

If e' has weight no lower than e, we can swap e in place of e' in T' to get $T'' = T' + e - e'$. Now $T' + e$ has a cycle containing e', and hence removing e' from $T' + e$ to get T'' does not disconnect it. Thus T'' is a spanning tree by the Spanning tree fact, and hence is an MST since its weight is no higher than that of T'. Also T'' contains $T + e$, because T does not contain e' and so is a subgraph of $T' - e'$. Therefore the invariance is preserved in this case as well.

Below is the formal proof corresponding to the above informal proof.

Theorem 5 (Kruskal's algorithm correctness).

For any connected weighted graph G:
Kruskal(G) is a minimum spanning tree of G.

Proof.

For any connected weighted graph G:
[We prove the invariant that T is always a subgraph of some MST of G.]
Let $w(e)$ be the weight of e, for any edge e in G.
During the execution of Kruskal(G):
T is set to be a graph with the same vertices as G but no edges.
Thus T is a subgraph of some MST of G [by Existence of MST].
For each edge e in G in order of increasing weight:
Post-invariance (to be proven):
T is a subgraph of some MST of G.
If $T + e$ has no cycle:
Let T' be some MST of G containing T [by post-invariance].
If e is in T':
$T + e$ is a subgraph of the T', which is an MST of G.
If e is not in T':
Note that T' contains a path connecting the endpoints of e [because T' is connected].
Thus $T' + e$ contains some cycle C.
Thus C has some edge e' that is not in $T + e$ [because $T + e$ has no cycle].
Also $T + e'$ does not contain a cycle [because T and e' are both contained in T' and T' is a tree].
If $w(e') < w(e)$:
There was an earlier iteration of the loop over e that considered e' during which:
Let T_0 be the version of T during that iteration.
T_0 is a subgraph of (the current) T.
Thus $T_0 + e'$ had no cycle [because $T + e'$ has no cycle].
Thus that iteration would have added e' to T_0.
Thus T should contain e', contradicting the choice of e'.
Therefore $w(e') \geq w(e)$.
Let $T'' = T' + e - e'$.
Then T'' is connected [because e' is in a cycle in $T' + e$].
Also the weight of T'' is at most the weight of T'.
Thus T'' is an MST of G [by the Spanning tree fact since T'' and T' have the same number of edges].
Thus $T + e$ is a subgraph of some MST of G [because T is contained in $T' - e'$].
Therefore $T + e$ is a subgraph of some MST of G.
T is set to $T + e$.
Thus the post-invariance is preserved.
Therefore T is a subgraph of some MST of G.
Thus the weight of T is at most the weight of an MST of G.
Also T is connected. [Why?]
Thus T is an MST of G.
T is returned.
Therefore Kruskal(G) is a minimum spanning tree of G.